AFRL-IF-RS-TR-2003-189
Final Technical Report
August 2003

JOINT BATTLESPACE INFOSPHERE (JBI) CORE SERVICES EVALUATION

Northrup Grumman Information Technology, Incorporated

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2003-189 has been reviewed and is approved for publication.

APPROVED:

/s/

PAUL T. WEBSTER
Project Engineer

FOR THE DIRECTOR:

/s/

RICHARD C. METZGER
JBI Program Manager
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	AUGUST 2003	Final Dec 01 – Dec 02

4. TITLE AND SUBTITLE
JOINT BATTLESPACE INFOSPHERE (JBI) CORE SERVICES EVALUATION

5. FUNDING NUMBERS
C - F30602-00-D-0159/0006
PE - 63789F
PR - JBIT
TA - PR
WU - O6

6. AUTHOR(S)
Stephen T. Scheiderich

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Northrup Grumman Information Technology, Incorporated
Defense Mission Systems
Beeches Technical Campus, Route 26 North
Rome New York 13440

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFSE
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2003-189

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Paul T. Webster/IFSE/(315) 330-3514/ Paul.Webster@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***
The focus of this work was to provide a methodology and process for evaluating JBI prototypes and providing feedback to the government for iterative development purposes. A methodology was developed and test plans generated for a generic JBI prototype evaluation from a functional point of view. Performance and scalability metrics were also identified as part of this effort. System engineering support (and documents) for the Mercury class of the JBI prototypes was also provided and is documented here.

14. SUBJECT TERMS
Joint Battlespace Infosphere, JBI, Publish and Subscribe, Test and Evaluation, Functional Testing, Performance Testing, Scalability Testing

15. NUMBER OF PAGES
59

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

INTRODUCTION

The JBI Core Services Evaluation task concerned itself with the methodology and performance of tests of candidate prototype 'JBI platforms' and subsequent evaluations in order to support an ongoing iterative development process instituted to provide the government with baseline and typical performance and capability requirements.

A number of tasks associated with this goal were identified in the statement of work as technical requirements or tasks to be performed. This document provides the results of this work using the statement of work as a template for presenting and demonstrating fulfillment of these requirements.

EVALUATION METHODOLOGY

As was required by the statement of work, a process and methodology were created to evaluate prototype JBI implementations and commercial off-the-shelf products. This methodology was created from a high-level perspective first, evolving into more specific processes as more information became (becomes) available for any one specific system under test. The overall or high-level methodology is documented in Figure 1 Test & Evaluation Process below.

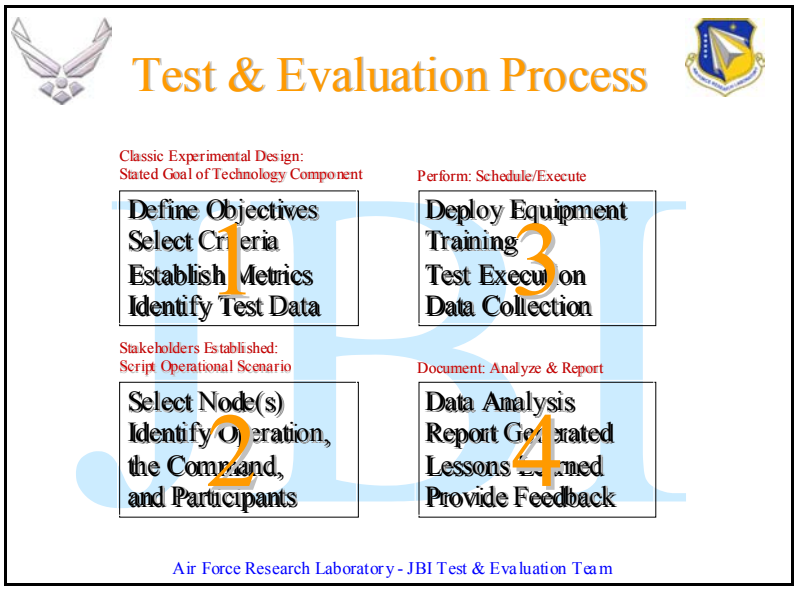

Figure 1: Test & Evaluation

This chart is derived from the process definition, which follows.

THE TEST & EVALUATION PROCESS

A methodology that can be used to provide information about systems in or under development that will spotlight outcomes based upon the focus of the tests. It has four main steps:

1. **Classic Experimental Design: Stated Goal of Technology Component**
   - Define Objectives
     - Establish the goals of the evaluation and testing.
   - Select Criteria
     - Establish identifying characteristics that are relevant to the goals (e.g., performance, scalability, and conformance to standards). Utilize formal system requirements if available.
   - Establish Metrics
     - Identify the measurements which will delineate the characteristics of the selected criteria.
   - Identify Test Data
     - Locate or create the data that will exercise the system under test that will result in measurements which highlight the characteristics desired.
2. **Stakeholders Established: Script Operational Scenario**
   - Select Node(s)
     - Identify the specific modules in the system under test for testing.
   - Identify Operation
     - Delineate the unit and its functionality that are to be the subject of the testing.
   - The Command
     - Locate or create an algorithm or procedure that will exercise the units' functionality.
   - And the Participants
     - Scale the testing procedure(s) and program(s) appropriately.
3. **Perform: Schedule & Execute**
   - Deploy Personnel & Equipment
     - Set up the hardware and software, as well as the personnel required to execute the tests. Set up the instrumentation to record the data.
   - Training
     - Inform, familiarize, and teach the test operators where necessary.
   - Test Execution
     - If all preparations have been successful, this is anticlimactic.
   - Data Collection
     - The procedures and operations required to store and provide the recorded data.

4. **Document: Analyze and Report**
   - Data Analysis
     - The process and procedures for reviewing, summarizing and aggregation of the data collected.
   - Report Generation
     - Creating the report(s) addressing the goals of the testing and evaluation.
   - Lessons Learned
     - A report detailing any failings and shortcomings of the process in order to improve it for future iterations.
   - Provide Feedback
     - Incorporate the test and evaluation and the lessons learned reports in order to provide information in regard to this process and its application for the particular system under test.

Document 1-1

SCOPE OF TESTING

This process was developed from industry best practices and experience acquired by the contract test personnel. This methodology is only as good as the personnel attempting to utilize it, as well as the quality of the inputs needed in order to execute it.

In order to provide the most useful evaluation to the government, the following document was distributed to ascertain the scope of the testing and evaluation (T&E) that was required in order to provide usable results to the government. These results were tabulated and averaged and then final results were used to provide direction for the T&E process.

Software Characteristics, their Testing and Evaluation

In order to evaluate the yJBIs being provided by vendors and to provide the most useful information for decisions about technical direction for this technology, it is necessary to state the level of testing required and the degree to which focus of a characteristic should be part of the overall test plan. (For example, Functionality should be 50%, Reliability should be 10%, Usability should be 10%, Maintainability should be 10%, and Portability should be 20%.) It is also very important to evaluate the level that a characteristic should be scrutinized, in order to maximize resources on evaluating what is important for the program goals. In *Table-1*, each level describes a level of testing to verify the characteristic where level A is the most exhaustive, and level D, the least.

In order to meet these goals, please fill out table-2 on the bottom of this page, referring where necessary to the *Table-1* for the descriptions and levels of testing for each characteristic.

Characteristic	Description	Level A	Level B	Level C	Level D
Functionality	A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are (typically) a capability to satisfy a stated or implied need.	Formal Proof	Functional Testing (Black Box)	Component Testing (Unit testing)	Review (Check List)
Reliability	A set of attributes that bear on the capability of software to maintain its level of functionality and performance under stated conditions for a stated period of time.	Formal Proof	Reliability growth model	Fault Tolerance Analysis	Programming Language Facilities
Usability	A set of attributes that bear on the effort needed for use, and on the individual evaluation of such use, by a stated or implied set of users.	Laboratory Testing	User Interface Inspection	Conformance to interface standards	User Mental Model
Efficiency	A set of attributes that bear on the relationship between the level of functionality of the software and the resources used to provide that functionality under stated conditions.	Performance Profiling Analysis	Execution Time Measurement	Benchmark Testing	Algorithmic complexity

Characteristic	Description	Level A	Level B	Level C	Level D
Maintainability	A set of attributes that bear on the effort needed to make specified modifications over time.	Traceability evaluation	Analysis of development process	Static Analysis	Inspection of Documents and Source code
Portability	A set of attributes that bear on the ability of the software to be transferred from one environment to another.	Program design evaluation	Environment constraints evaluation	Conformity to Programming Rules	Analysis of Installation
Performance	A set of attributes that bear on the ability of the software to provide timely responses for the stated functions for the stated conditions	Performance Profiling Analysis	End-to-end User Response Testing	Algorithm and Code Evaluation	Application Characterization

Table –1: Software Characteristics, Descriptions and Levels of Testing

Please fill out *Table-2* on this page, referring where necessary to *Table-1* for the descriptions and levels of testing for each characteristic.

Name	Email	Phone

Characteristic	Percentage of Focus	Level of Testing
Functionality		
Reliability		
Usability		
Efficiency		
Maintainability		
Portability		
Performance		

Table-2

Document 1-2

CHARACTERISTICS OF A YJBI

In order to create a test plan that identifies specific functionality, explicit performance, and precise security requirements, each of these areas needed to be clearly documented within a software specification document. Since a document of this nature was not available, the test team set about to identify basic functionality and performance requirements in order to establish a 'generic test plan'. First, a characteristics document was created which attempted to identify basic functionality. (This can be seen as document 1-3 below.)

Characteristics Document

The Platform Service is made up of these functions:

1. brokering
2. control
3. information
4. dissemination
5. fuselet management
6. and simple transformation utilities

Test cases for the above primary functionality:

Platform provides:

BROKERING

- (Resource, security, and policy constraint management to clients)
- Registration services to clients/fuselets
- Subscription add service to clients/fuselets
- Subscription modify service to clients/fuselets
- Subscription delete service to clients/fuselets
- Querying service of published objects to clients/fuselets (only metadata)
- Status information about subscriptions/queries to the client/fuselet
- Publishing an information object(s) service in a repository (metadata and payload)
- Publishing an information object(s) service (just metadata, persist IO at publisher)
- Retrieving information object(s) service in a repository (metadata and payload)
- Retrieving information object(s) service from metadata on platform from publisher
- Remove information object(s) service from a repository
- Remove information object(s) service metadata from a repository (payload at publisher)
- Peer information for publisher to subscriber connection(s) (p2p)
- Notify subscriber of InfoObjects meeting requirements (push InfoObject list)
- 'Pushes' subscribed and selected InfoObjects metadata to subscribers from repository
- 'Pushes' subscribed and selected InfoObjects(whole) to subscribers from repository

- ‘Pushes’ information objects to subscribers from metadata on platform with payload from publisher
- ‘Pushes’ information objects to subscribers from publisher (unicast or broadcast or multicast?)
- Persistence for an information object published to a repository
- Retrieving information object(s) schemas
- Adding information object(s) schemas
- Editing information object(s) schemas
- Deleting information object(s) schemas
- Searching information object(s) schemas

CONTROL

- Platform provides login/logout for users through clients (includes authentication)
- Platform provides connection functions for clients, fuselets, and brokers (with authentication)
- Platform registration (login)
- Platform de-registration (unsubscribe and logoff)
- Platform tracks client/fuselet/other broker status (connection and otherwise)
- Platform adds subscription
- Platform modifies subscription
- Platform deletes subscription
- Platform provides control services to ‘distribute’ platform services to multiple systems. (tbd)
- Platform provides control services to maintain/manage platform databases (Repositories)
- Add/Maintain/Remove/Archive database schemas, indexes, stored procedures, row triggers, etc.
- Add/Maintain/Remove participating repository storage systems
- Provide control services for the SAN (Storage Area Network) behind the JBI
- Platform provides for editing system (JBI Platform) settings
- Platform provides for editing security settings (permissions)
- Platform identifies any other connected brokers and subscribes to their publications (where appropriate)
- Platform identifies all other connected brokers and adds their subscriptions
- Platform identifies and adjusts content depending on destination bandwidth capability and other factors
- Platform maintains a log (or log files) of all interactions with it (security auditing)
- Platform maintains performance metric(s) information object(s) which are being updated continuously (QoS)
- Platform maintains system status information object(s) which are being updated continuously (QoS)

- Platform identifies high load levels and shifts (some or all) operations to other systems providing platform services (QoS)
- Platform provides for controlling of network connectivity relating to QoS settings
- Platform provides for retrieving information object(s) schemas
- Platform provides for adding information object(s) schemas
- Platform provides for editing information object(s) schemas
- Platform provides for deleting information object(s) schemas
- Platform provides for searching information object(s) schemas

INFORMATION

- Stores/retrieves/maintains list of information object(s) schemas (and associated types)
- Stores/retrieves/maintains list of publishers
- Stores/retrieves/maintains list of subscribers
- Stores/retrieves/maintains list of subscriptions
- Stores/retrieves/maintains list of fuselets
- Stores/retrieves/maintains list of force templates
- Stores/retrieves/maintains list of brokers (and associated systems providing platform services)
- Stores/retrieves/maintains list of persisted information objects
- Provides information relating to operations (pub/sub/query) performance
- Provides information relating to network connectivity and latency

DISSEMINATION

- Provide querying capability to <all known> information objects
- Provide browsing capability for <all known> information objects
- Provide transmission capability for <all known> information objects

FUSELET MANAGEMENT

- Platform creates fuselet(s)
- Platform modifies/edits fuselet(s)
- Platform deletes fuselet(s)
- Platform modifies/edits fuselet permissions
- Platform advertises fuselet(s)
- Platform publishes fuselet(s)
- Platform executes fuselet(s)
- Platform delivers fuselet(s)
- Client(user) adds fuselet to platform
- Client(user) modifies fuselet(s)

- Client(user) deletes fuselet(s)
- Client(user) modifies/edits fuselet permissions
- Client(user) executes fuselet(s)

TRANSFORMATION UTILITIES

- Translator creates fuselet
- Translator modifies/edits fuselet
- Translator deletes fuselet
- Translator modifies/edits fuselet permissions
- Translator executes fuselets for specified conditions

Document 1-3

DATA FLOW DIAGRAMS

The characteristics document was followed by data-flow diagrams whose purpose it was to identify functions which would be subject to testing via black-box methods (from the outside). These can be seen in figures two (2) through eight (8) below.

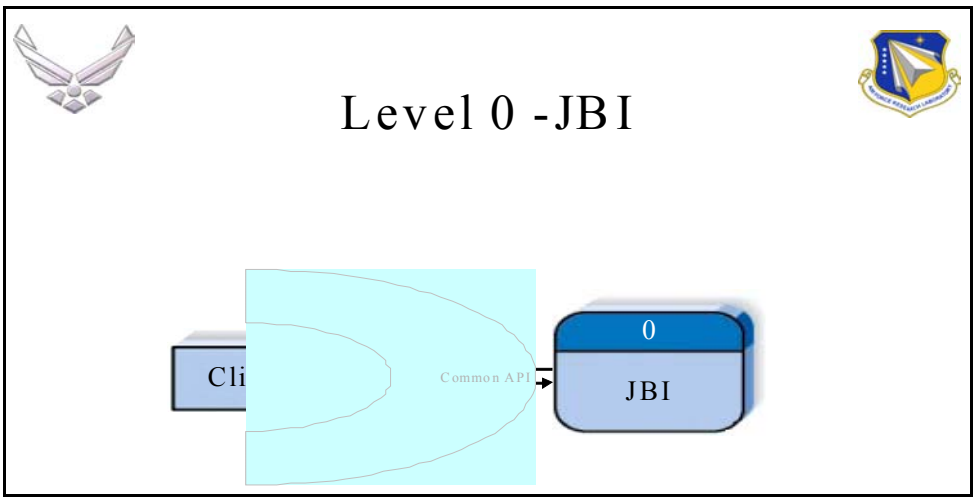

Figure 2: Data Flow Diagram Level 0

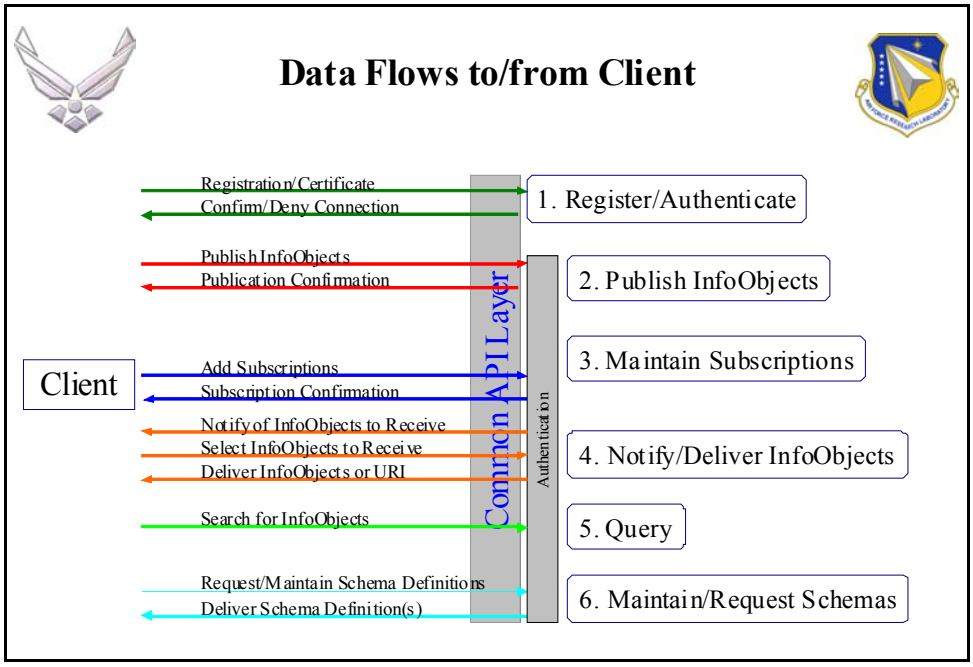

Figure 3: Data Flow to/from Client

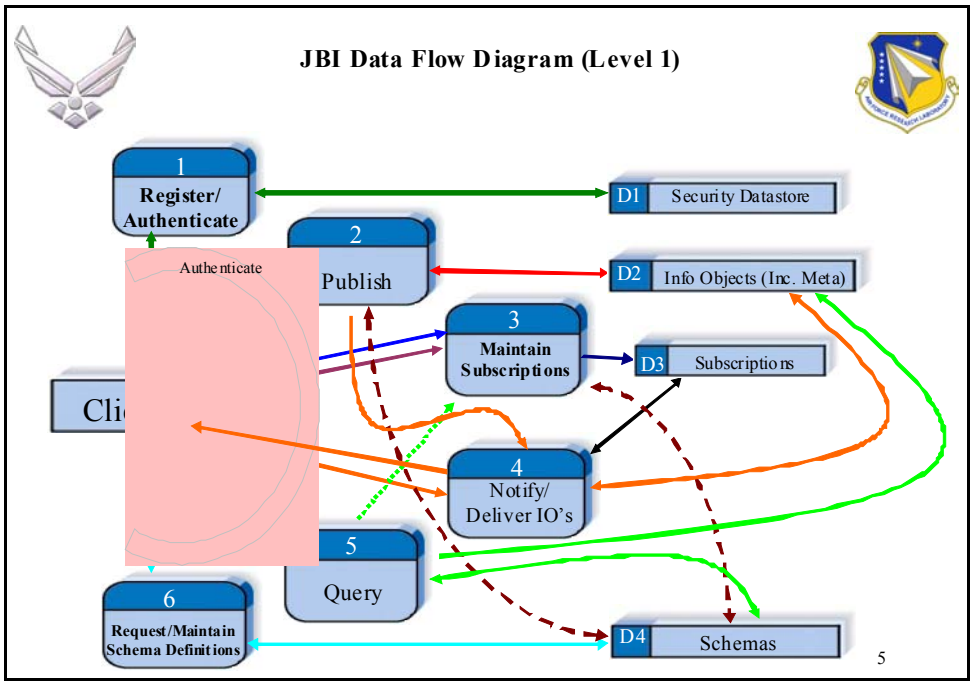

Figure 4: Data Flow Level 1 Diagram

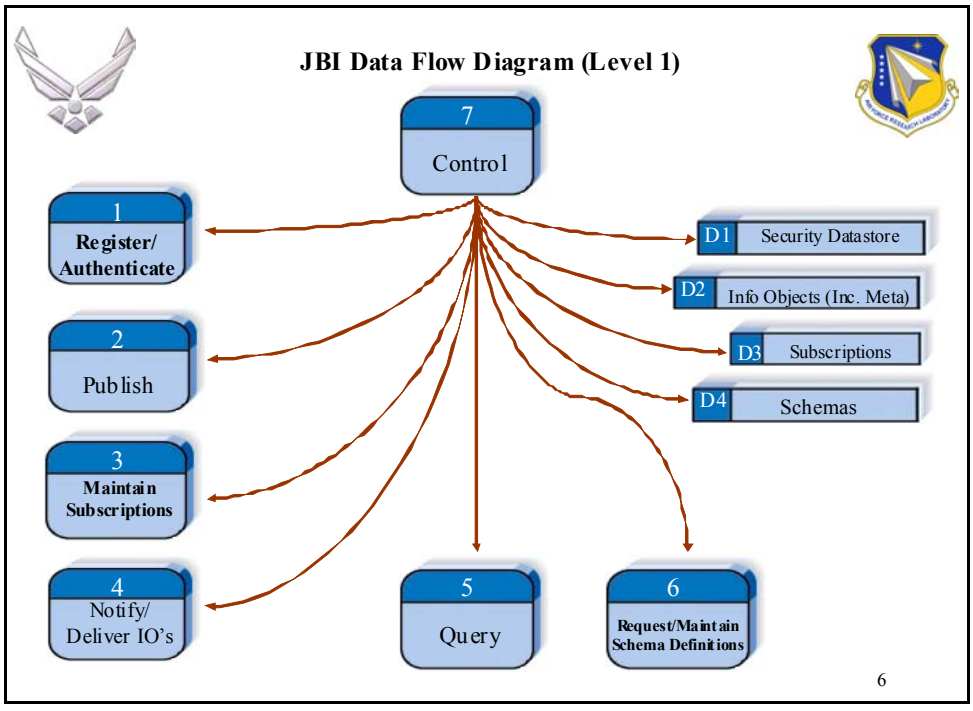

Figure 5: Data Flow Diagram Level 1

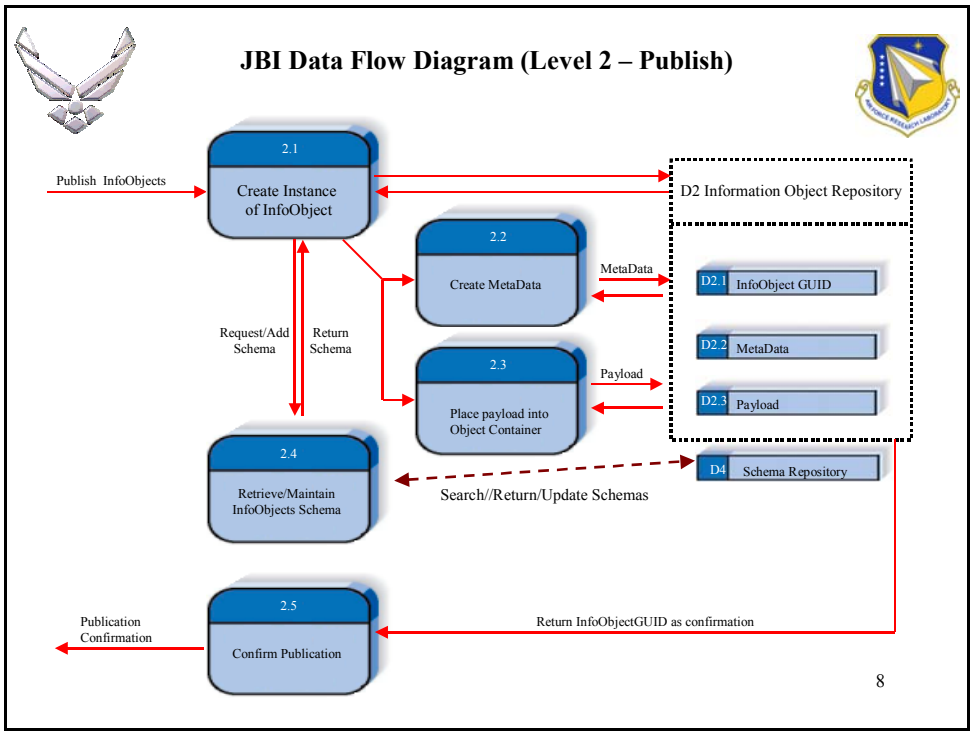

Figure 6: Data Flow Diagram Level 2 – Publish

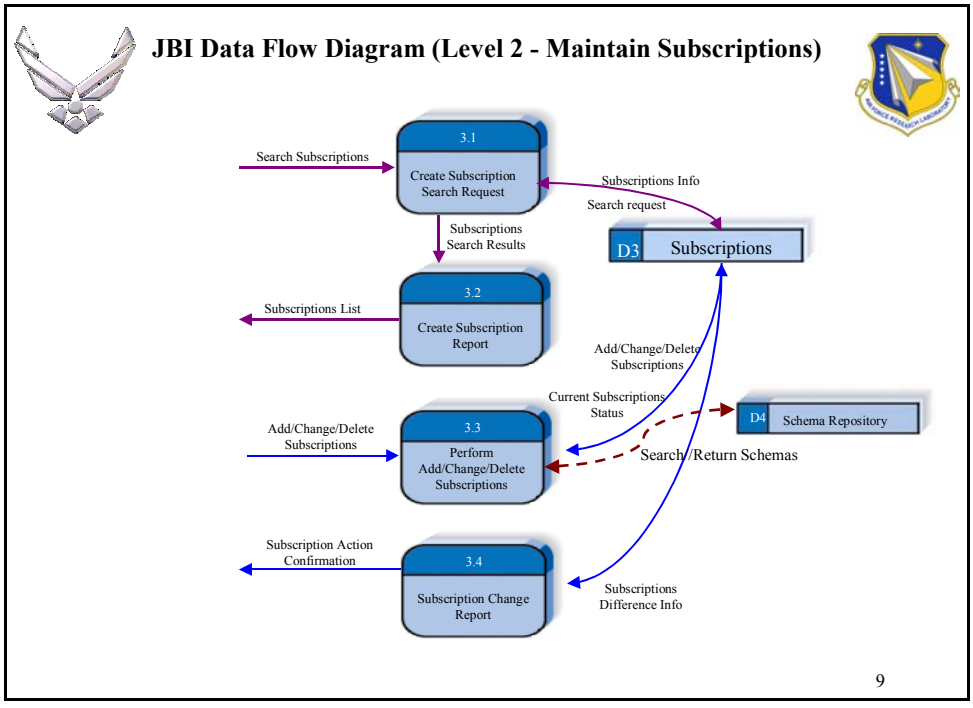

Figure 7: Data Flow Diagram Level 2 – Subscription

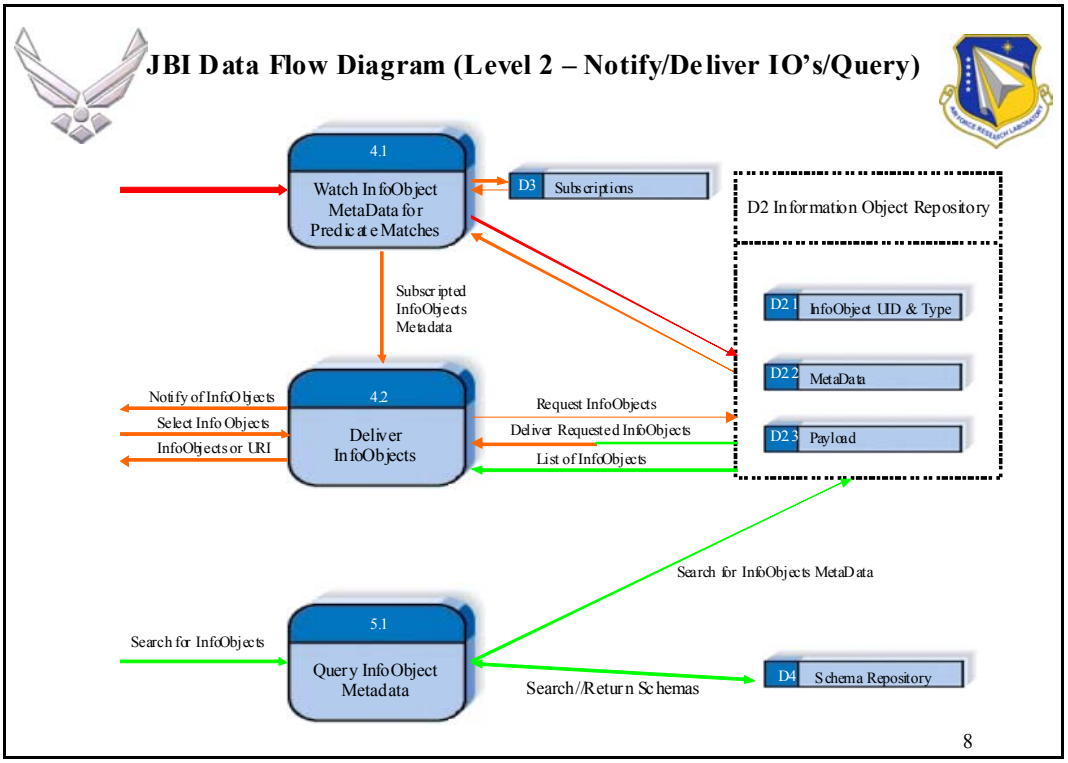

Figure 8: Data Flow Diagram Level 2 – Notify/Query

The test team also worked on identifying capability as inferred from the Common Client API (Application Programming Interface), usually called just the Common API. To that end, data-flow diagrams for the Common API were also completed in order to provide insight into the operational capability from a 'black box' perspective. These diagrams are figures nine (9) and ten (10) below.

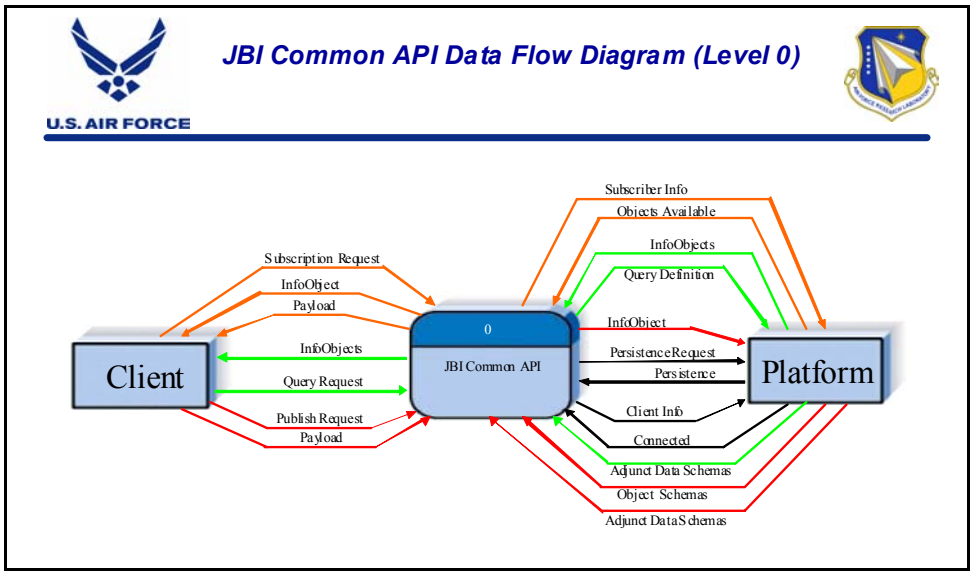

Figure 9: Common API Data Flow Diagram Level 0

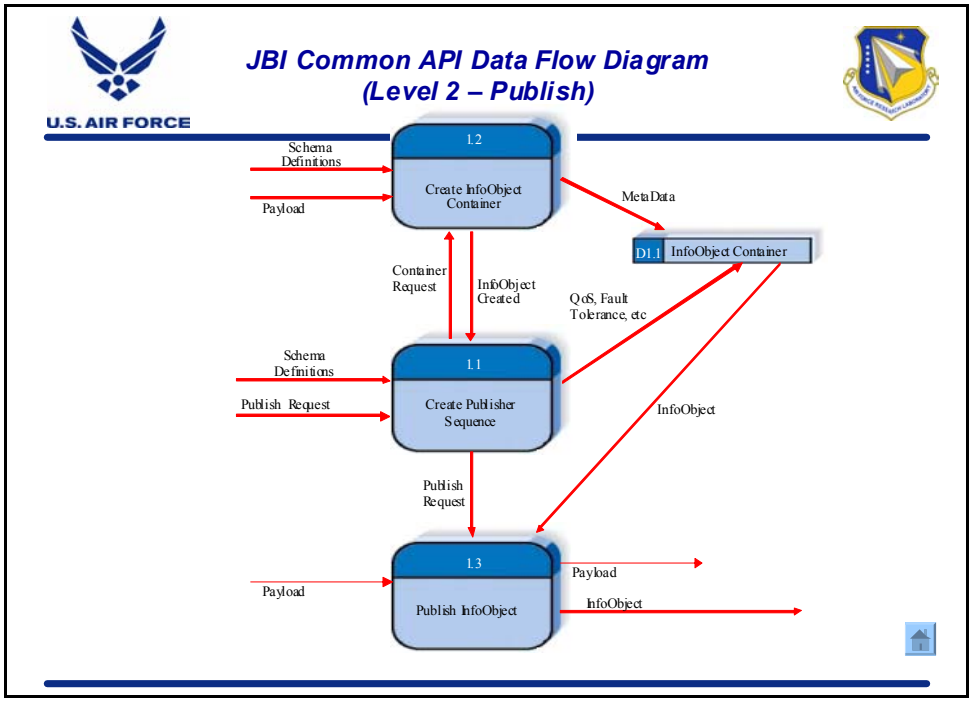

Figure 10: Common API Data Flow Diagram Level 2

All of these data-flow diagrams utilized the Gane-Sarson symbols for processes and data sources.

USE CASE DIAGRAMS

As part of the process of identifying functionality, the system engineering team and test team created UML (Unified Modeling Language) use cases for the JBI Platform from the perspective of the Common API. These can be seen as figures 11 through 16. None of these diagrams or the descriptions that proceed from them are to be considered formal in any way. The goal of this work was to identify functionality in order to provide a basis for test plans predicated on expected functionality. When formal requirements are identified by the government, then the test plans will be revised to reflect the differences.

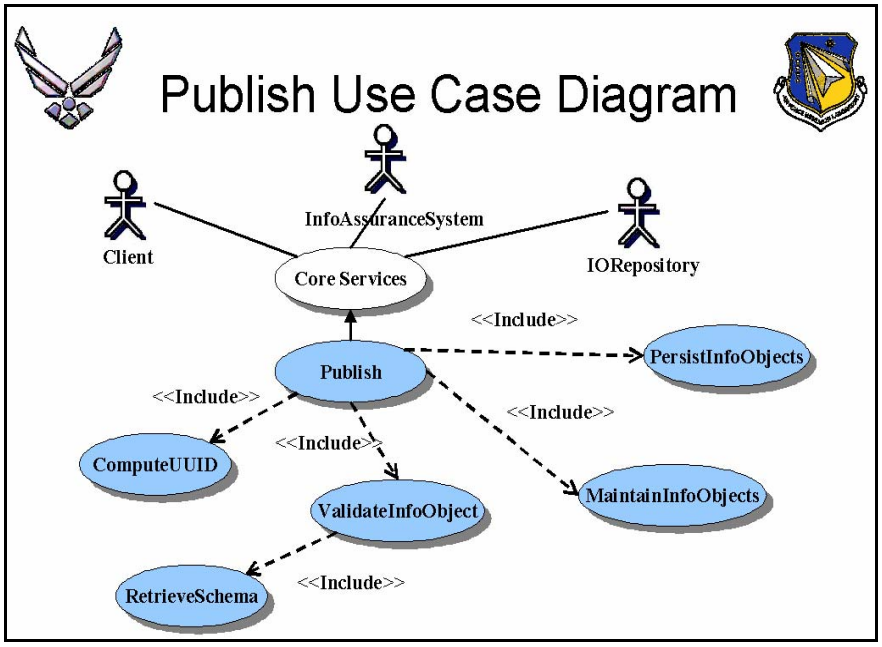

Figure 11: Publish Use Case

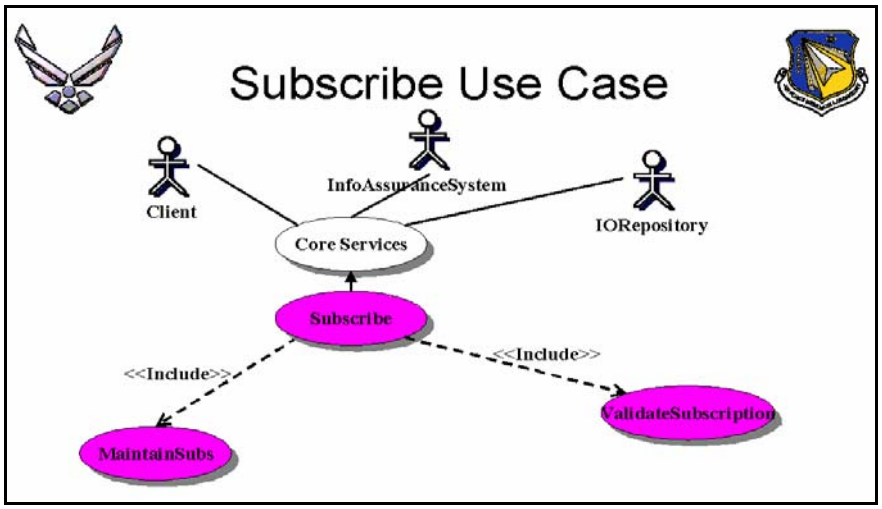

Figure 12: Subscribe Use Case

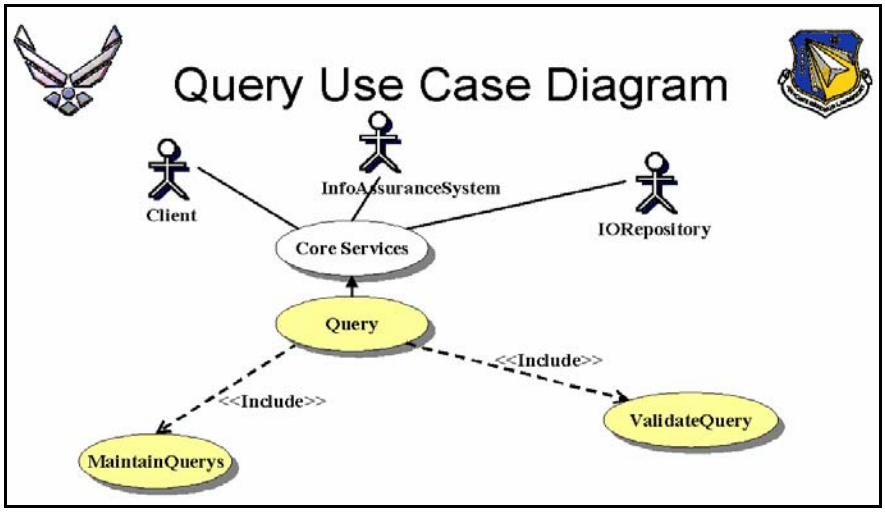

Figure 13: Query Use Case

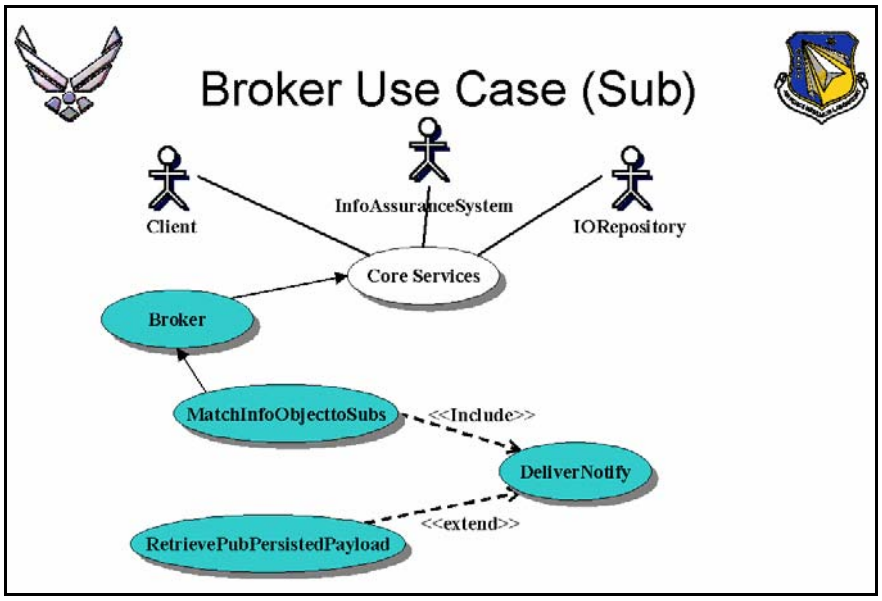

Figure 14: Broker Use Case (Subscription)

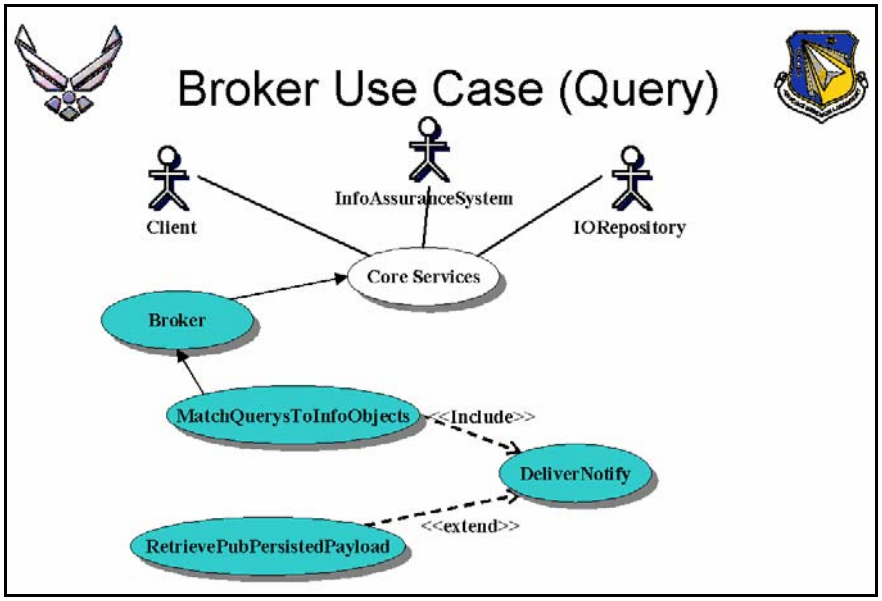

Figure 15: Broker Use Case (Query)

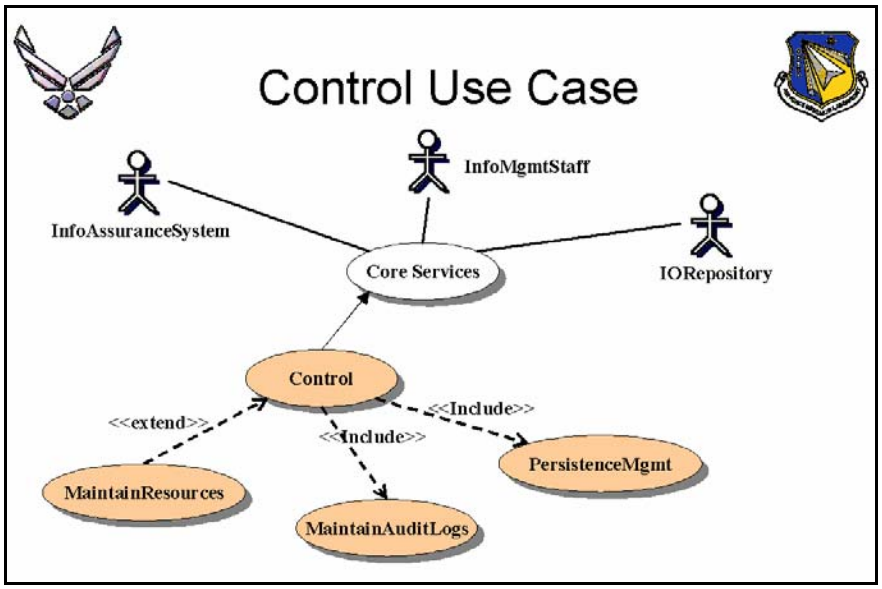

Figure 16: Control Use Case

FUNCTIONAL REQUIREMENTS

These use case scenarios identified functionality in regard to client versus server and provided a refinement of the identified functionality which was further documented as below in Document 1-4.

Specific Functional Requirements of a JBI

- The ability to 'publish' information objects to an infosphere. These would also be placed in a repository.
- The ability to deliver information objects that fulfill a request (subscription) as they are published.
- The ability to deliver information objects that fulfill a request (query) from an existing published information objects collection (repository)
- The ability to prioritize these functions by client, by information object content, by request, etc., based on a policy established by the JBI commander and affected by current operational parameters (including bandwidth capabilities as well as situational conditions)
- The ability to search for, and retrieve schemas in support of the three primary functions(publish, subscribe, & query).
- The ability to identify a JBI information object using a unique identifier. (UID)
- The ability to use XML as the fundamental Information Object structure architecture.
- The ability to use XML Schema (only) to define Information Object Types.
- The ability to validate that an information object conforms to a Schema.
- The ability to validate that an information object is well formed. (XML and Schema)
- The ability to control the three primary functionalities by client/user, by information object, by request, etc., and the resources needed to provide the three primary functionalities.
- The ability to retrieve from and use legacy data from in-place C2 systems.
- The ability to update legacy C2 systems with new data where relevant.
- The ability to federate JBIs together where information can be shared between JBIs so that information objects in one JBI can be subscribed to and queried from another.
- The ability to stand-up a JBI in a relatively quick way, where JBI repository is already populated with relevant data for areas of interest (AOIs).
- The ability for a military unit (squadron for example) to be provided with connectivity which supports full functionality in terms of the expected unit operational capabilities for both receiving data (orders, tactical/strategic data, etc.,) and sending/responding to requests/orders for data (reports, results, etc.,) (FORCE TEMPLATE concept)
- The ability to create and use small programs (FUSELETS) to subscribe to, manipulate, and publish information objects both in automated ways as well as interactively to derive/provide knowledge from existing information objects.
- The ability to create and use small programs (FUSELETS) to 'transform' data from one form to another, where the being converted can be coming from a JBI and being sent out of the JBI, to incoming data being converted into information objects for the JBI.

- The ability to control, store, and create small programs (FUSELETS), which provide the 'glue' for the main capabilities of the JBI.
- The ability to maintain proper information assurance for both the systems and the data that make up the JBI(s).
- The ability to authorize and authenticate individual users and also to authorize by role individual users which are identified as part of a military unit instantiated through the "Force Template" mechanism.
- The ability to control individual information dissemination mechanisms based upon client and user authorization (need to know).

Document 1-4

BUILDING A TEST LABORATORY

Recommendations for the hardware/software infrastructure components to be acquired for the JBI Distributed Test bed (JBI-DT) were provided as follows:

Building a Test Laboratory

- Flexibility is the goal
- Try to accommodate all possible infrastructure requirements for the testing which is to be done
- Try to set aside enough lab space for current requirements and accommodate future growth including moving the test lab (in effect, have plug and play for the lab)
- Create demarcation points for incoming and outgoing electrical connections, i.e., Power(120V & 220V) and Network(T1/T3, Local LAN/WAN, etc.)
- Locate a wiring rack/closet near the above point that allows connections to be changed for test conditions
- Provide modular/flexible connections for all (work)stations in the lab

Specifics

- Distribute sufficient power to each (work)station for anticipated use (at least six (6) outlets with each station on a 15A/20A breaker)
- Provide four (4) network drops to each (work)station (possible more) which are traceable back to a jack-panel in the wiring rack/closet
- Four network jacks at each station as such:
  1. internet (via proxy)
  2. test lab private LAN (Fast/Ethernet)
  3. test lab private LAN (Fast/Ethernet)
  4. test lab private LAN (ATM switched)
- two telephone jacks at each station
  1. regular phone connection
  2. test lab phone line
- Wiring Closet/rack contents:
  1. Router/Proxy connected to RRS Lab backbone with DHCP for internet access
  2. Isolation switch for instantaneous disconnection from the internet through router/proxy
  3. Level 2/3/4 Switches for the private Testbed LAN
  4. Wireless Switch/Hub to simulate real-world conditions.
  5. Jack panels with jacks for ALL (work)stations in the lab with extra capacity for growth
- Physical connectivity:
  1. All network wire running in modular cable trays below ceiling which will provide maximum flexibility for adding or moving connections, as well as the trays themselves
  2. Network wire color-coded and bunched (velcro straps) by destination to indicate type of connection

3. Establish a File-Server for the test lab
4. All software to undergo testing
5. Data results from the testing

- All documentation and reports (before CMDB)
- Can also be a Print-Server for the test lab
- Backups maintained on this system (Tape/CDRW capability)
- Has potential dual network connectivity (test lab/internet through proxy)
- Establish a test lab file-server to maintain local copy of patches, hot-fixes, service packs, and OS versions (Solaris, WinNT/2k+, Linux)
- Store Open Source and (GNU and others) software packages used by the test lab here

Document 1-5

A GENERIC yJBI TEST PLAN

The results of this work collectively allowed the test team to create the following draft test plan for functionally testing a yJBI prototype. It is not considered to be complete or necessarily appropriate for any one prototype, but it does represent a distillation of the expected functionality of a JBI platform as derived from the SAB report and the JBI test team. Actual testing of JBI prototypes would use this test plan as a starting point for functional testing in the absence of an actual requirements specification that could be used for acceptance (and functional) testing.

Generic yJBI Test Plan

Document Change Record

Table1. Document Change Record

1. Document Title: Generic yJBI Functional Test Plan			
2. Document Reference Number		JBI Test Plan	
4. Issue	**5. Revision**	**6. Date**	**7. Reason for Change**
0	1	2002-3-25	Start
0	1.1	2002-4-18	Reviewed/Updated by S. Scheiderich
0	1.2	2002-4-19	Amended by D. Rash
0	2.0	2002-4-19	1st Formal Draft
0	2.1	2002-5-14	Post Comment Update by D. Rash
0	2.2	2002-10-12	Updated by S Scheiderich

The basis of this outline has been taken from the IEEE Standard for Software Test Documentation (IEEE Std 829-1983 from IEEE Standard for Software Test Documentation) and it has been modified for the testing environment of the yJBI Functional Testing for the JBI Distributed Test Bed.

1.0 Introduction

The test plan is made directly from the Function List which was derived from the following reports: Building the Joint Battlespace Infosphere (SAB-TR-99-02), JBI Reference Architecture, and the JBI FAQ. Some functions were added on from the Dolphin Technologies JBI Requirements Report. This test plan will undergo significant development as the formal design documents of the JBI evolve. The function list that was used here was derived from a generic data-flow diagram (Gane-Sarson Model) that was created by the Test team for this purpose.

The test plan is divided into functional and performance testing with parts of performance testing verifying client reporting capabilities.

2.0 Scope

Functional testing will be applied to a yJBI prototype using the following test plan. Changes to the test plan scope to accommodate different yJBI implementations are expected and need to be

accommodated when estimating hardware/software requirements, as well as set-up time for the tests.

Functional testing does not test for the ability of the item under test (IUT) to perform correctly under all circumstances. For this kind of testing Reliability tests must be performed. Functional testing also does not ensure that when a function does not work correctly, or other parts are incapacitated that the trouble is easily diagnosed or easily repaired. The type of testing more acclimated to those findings is testability and maintainability, respectively. Functional testing can determine the presence and functionality of reliability, testability, and/or maintainability if these are functional requirements of the IUT. For instance, the SAB clearly documents the requirement for "Measurability and manageability: the commander must be aware of the status of the JBI. Problems need to be quickly identified and repaired." (Page 93), which assumes some sort of testability and/or maintainability functions are built into the JBI.

3.0 Test Items

Functional testing is intended to include an assessment of the interplay between components as well as test for individual actions. The tests are developed with the following questions in mind: Are all requirements being met? Are all functions working as intended? Do all functions startup and shutdown correctly?

The following are the test steps to test the functionality of a generic yJBI prototype.

Brokering Functions

Registration Service To Clients

Purpose: If the clients are required to identify themselves to the yJBI, test for this functionality. If this connectivity is established through a user-login, then use this to test the yJBI for this functionality.

Test Steps:

1. Establish a connection to the yJBI from a client if the yJBI supports this capability. If this requires logging-in a user, then test for this functionality. If it does not require a user, then establish the connection without a user login.
2. If, during the process of connecting, the client is required to provide a 'user profile' or some such information in order to complete the connection, identify, test and document this requirement.
3. Document this functionality (or lack thereof).

Subscription Add Service

Purpose: Once connected to the yJBI, the client adds a subscription for information objects that may be published in the future.

Test Steps:

1. Connect to the yJBI (if not already connected).
2. Create a subscription (or request, whatever method is required for subscriptions in this yJBI) and submit it to the yJBI for fulfillment when information objects become available that meet the requirements of the subscription. Identify whether or not the subscription was recorded in the appropriate subscription repository (database). If a metadata schema was/is required for the subscription, identify whether or not the metadata schema repository was queried for and returned the proper schema information.
3. During the process of subscribing, identify, test and document each of the various possible permutations of subscription attributes.
4. Document this functionality (or lack thereof).

Subscription Modification Service

Purpose: To determine if the client is able to retrieve, modify, and re-save a subscription back to the platform.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software
2. Retrieve an existing subscription.
3. Modify this subscription, (retrieving schema information as well, potentially), use all possible permutations of subscription attributes to test this capability.
4. Save and implement this (these) modified subscription(s).
5. Document this functionality (or lack thereof).

Subscription Delete Service

Purpose: To delete a subscription from the system.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software
2. Retrieve or select an existing subscription.
3. Delete this subscription.
4. Determine if the subscriber receives a message indicating subscription loss.
5. Verify the Subscription has disappeared from the list of subscribers.
6. Document this functionality (or lack thereof).

Status Information About Subscription/Queries To Client

Purpose: To determine if a client can be given status information from the Y-JBI.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.
2. Execute any steps that might be necessary to retrieve status information about queries or subscriptions which have been instituted.
3. Determine if status messages appear anywhere regarding subscription or query status.
4. Document this functionality (or lack thereof).

Querying Of Published Objects

Purpose: To determine if objects can be queried from the Y-JBI.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.
2. Execute any steps that might be necessary to query for information objects in the yJBI. (Validate that information objects are available to be queried from the yJBI independently.)
3. Select and retrieve information objects identified in step 2.
4. Document this functionality (or lack thereof).

Publishing Of Information Objects (Add Publications)

Purpose: To verify that a client can publish an information object into the yJBI and that the published object can be subscribed and queried from the yJBI.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.
2. Execute any steps that might be necessary to publish a new information object.
3. Record whether the yJBI provides confirmation that the object was published.
4. Verify independently that the object was published by receipt of the object via a subscription as well as query after the fact.
5. Document this functionality (or lack thereof).

Publishing Of Information Objects Persisted By The Publisher

Purpose: To verify that a client can publish an information object and provide persistence for that object as well as delivering that object to any subscribers through the yJBI that are identified by the JBI platform (where the metadata for this information object has been stored) to this client.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.
2. Execute any steps that might be necessary to publish a new information object and persist its existence on the client.
3. Record whether the yJBI provides confirmation that the object was published.
4. Verify independently that the object was published by receipt of the object via a subscription as well as query after the fact. Verify that the object is NOT on the yJBI platform except for its metadata and that the object can be queried through the yJBI from other clients where the object appears to be 'on' the platform but is not.
5. Document this functionality (or lack thereof).

Publishing Modify Service

Purpose: To determine if the platform allows published object to be modified.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.
2. Execute any steps required to Query for a published object.
3. Select an appropriate object and attempt to modify the object. (Modification of an object may entail new versions, where the original object cannot be changed but where subsequent 'versions' can be published which effectively replace the original object.)
4. Record whether the yJBI provides confirmation that the object was published.
5. Verify independently that the object was published by receipt of the object via a subscription as well as query after the fact.
6. Document this functionality (or lack thereof).

Publishing Delete Service

Purpose: To determine if the platform allows a published object to be deleted from the platform.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.
2. Execute any steps required to Query for a published object.
3. Select an appropriate object and attempt to delete the object. (Deletion of an object may entail multiple versions, where the original object plus subsequent 'versions' could be deleted.) Test for both possibilities.
4. Record whether the yJBI provides confirmation that the object was deleted.
5. Verify independently that the object was deleted by the lack of receipt of the object via a subscription as well as query after the fact.
6. Document this functionality (or lack thereof).

Notify Subscriber Of Information Objects Meeting Requirements

Purpose: To verify that the platform can and does deliver notifications of Information Objects that meets a subscribing/query request.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.
2. Execute any steps required to Query for a published object.
3. Execute any steps required to Subscribe to an unpublished object.
4. Publish, from another client an information object meeting both of these requests.
5. Verify the subscriber and query clients received a notice of published information objects based on their requests.
6. Document this functionality (or lack thereof).

Retrieving Information Object Schemas

Purpose: Test the ability to retrieve Information Object Schemas.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.
2. Execute any steps required to query and retrieve any information object schemas (if able).
3. Document this functionality (or lack thereof).

Adding Information Object Schemas

Purpose: To test the ability for adding Information Object Schemas.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.
2. Execute any steps required to add Information Object schemas (if possible).
3. Validate that this operation has taken place independently of the process under test.
4. Document this functionality (or lack thereof).

Modifying Information Object Schemas

Purpose: To test the ability to modify Information Object Schemas.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.

2. Execute any steps required to query, retrieve and modify Information Object Schemas.
3. Validate that this operation has taken place independently of the process under test.
4. Document this functionality (or lack thereof).

Deleting Information Object Schemas

Purpose: To test the ability to delete Information Object Schemas.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.
2. Execute any steps required to query, retrieve and delete Information Object Schemas.
3. Validate that this operation has taken place independently of the process under test.
4. Document this functionality (or lack thereof).

Control Functions

Login/Logout For Users Through Clients

Purpose: To test the login and logout capabilities of the system under test. Establish the functionality of the client versus the system. Identify the characteristics of the system providing the service(s) of login and logout.

Test Steps:

1. Verify that a login and logout are provided by some mechanism. Use this mechanism to login and maintain this condition until a timeout occurs.
2. Validate that a login has occurred by recognition of the user by the yJBI control services (if available).
3. Once logged in, determine if the client is able to logout of the yJBI.
4. Validate that a logout has occurred through the yJBI control services.
5. Document this functionality (or lack thereof).

Provide Connection Functions For Clients

Purpose: To test the capability (if it exists) for a client program to connect with and use JBI platform services without a user. (Might typically be representative of a fuselet connecting to and using platform services.)

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client (fuselet) software without user intervention other than test initialization and providing whatever trigger is necessary (artificially) to execute the program (fuselet).

2. Verify and validate that this process was successful by viewing published information objects created by this program (fuselet) with another client.
3. Document this functionality (or lack thereof).

Registration

Purpose: To determine if the platform provides a method of registration and to test this capability. Registration is seen as the user or client providing what has been referred to as a 'profile' of the user's anticipated interaction with the yJBI. This quite probably will be in the form of data indicating the scope of interactions (publishing, subscribing, querying), as well as (possibly) specific client/user requirements in terms of hardware attributes (IP address, hostname, interface, throughput, display capabilities, etc.), and other miscellaneous information (such as specific predicate language) as yet undefined.

Test Steps:

1. Establish the registration document/file, which the software will recognize (given documentation establishing this).
2. Determine if the client is required to log in directly using separate subscribe, query or publish client programs, if so, then there is no separate registration process.
3. If there is a separate registration step, connect to the yJBI (if not already connected) with provided client software.
4. Execute any steps required to login, and provide the registration document.
5. Validate that this operation has taken place independently of the process under test.
6. Document this functionality (or lack thereof).

De-Registration

Purpose: To determine if the yJBI properly disassociates the provided 'profile' from the client/user when the de-registration process is executed if it has this capability.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.
2. Execute any steps required to login, and provide the registration document (if not already done).
3. Execute any steps required to de-register by removing the registration document (the 'profile').
4. Validate that this operation has taken place independently of the process under test.
5. Document this functionality (or lack thereof).

Check Subscription Broker Status

Purpose: Test for the capability to monitor the Subscription Broker status using the control services interface to the yJBI (If there is one). Alternately, to test for the capability to monitor the process which handles the subscriptions.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Execute any steps required to monitor the subscription functions (broker per se)
3. Document this functionality (or lack thereof).

Provide Control Services To Maintain/Manage Platform Databases (Repositories)

Purpose: Test to determine if the platform allows the manipulation of repositories.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Execute any steps required to access, monitor, or modify the repositories.
3. Document this functionality (or lack thereof).

Provide For Editing System (JBI Platform) Settings

Purpose: To determine if the platform allows editing of JBI platform settings.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Execute any steps required to access, monitor, or modify Platform settings.
3. Document this functionality (or lack thereof).

Provide For Editing Security Settings (Permissions)

Purpose: To determine if the platform allows editing of security settings.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Execute any steps required to access, monitor, or modify security settings for the various entities (Objects).
3. Document this functionality (or lack thereof).

Identifies Any Other Connected Brokers And Subscribes To Their Publications (Where Appropriate)

Purpose: To determine if the current JBI platform can 'see' other JBI platforms and interact with them in terms of the publish, subscribe, query paradigm.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Execute any steps required to view, connect to, interact with other JBIs.
3. Document this functionality (or lack thereof).

Identifies And Adjusts Content Depending On Destination Bandwidth Capability And Other Factors

Purpose: To determine if the JBI platform can identify and adjust content based on current network environmental factors.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. See if the control interface can identify network environmental properties such as latency, bandwidth available, etc.
3. See if the control application can adjust delivery rates of any published objects.
4. Document this functionality (or lack thereof).

Maintains A Log (Or Log Files) Of All Interactions With It. (Security Auditing)

Purpose: To determine if the platform keeps track of all interactions via examinations of log files or records.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Determine if the system keeps logs of activity for the various types of interactions and can view them and annotate them.
3. Document this functionality (or lack thereof).

Maintains Performance Metric(s) Information Object(s) (Logs) Which Are Being Updated Continuously (QoS)

Purpose: To determine if the platform can access log files of platform metrics.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.

2. Determine if the system keeps logs of performance metrics for its various subsystems and whether they can be viewed or annotated.
3. Document this functionality (or lack thereof).

Maintains System Status Information Object(s) (Logs) Which Are Being Updated Continuously (QoS) Identifies High Load Levels And Shifts (Some Or All) Operations To Other Systems Providing Platform Services (QoS)

Purpose: To determine if the platform maintains system status that report high load levels and redistributes loads automatically.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Determine if the system keeps logs of system status for its various subsystems and whether they can be viewed or annotated or controlled.
3. Document this functionality (or lack thereof).

Provides For Control Of Security On A Need-To-Know Basis Settings

Purpose: To determine if the platform can control security settings within classification levels, such as need-to-know within a top secret setting or if this capability is implemented in a separate layer.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Open a subscriber/query client application.
3. Select an Information Object and change the security setting so the subscription or query can no longer access the Information Object.
4. Determine if a query can access the Information Object.
5. Document this functionality (or lack thereof).

Provides For Controlling Of Network Connectivity Relating To QoS Settings

Purpose: To determine if the platform can control Information Object exchange rates.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Determine if the system has the ability to adjust data exchange rates for information objects and test this capability.
3. Document this functionality (or lack thereof).

Information Functions

Stores/Retrieves/Maintains List Of Information Object(s) Schemas (And Associated Types)

Purpose: To determine if the platform maintains a list of Information Object schemas and metadata types and can control these through the administrative control interface. (Add, change, update, delete, etc.)

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Determine if the system has the ability to add, modify, view and delete information object schemas and test this capability.
3. Document this functionality (or lack thereof).

Stores/Retrieves/Maintains List Of Persisting Publishers

Purpose: To determine if the platform maintains a list of publishers for control and measurement and if it is possible to manage these (e.g., move published object onto platform).

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Determine if the system has the ability to add, modify, view and delete publishers who have persisted objects and whether it is possible for the platform to move or replicate these objects.
3. Document this functionality (or lack thereof).

Stores/Retrieves/Maintains List Of Subscribers

Purpose: To determine if the platform maintains a list of subscribers and their subscriptions for control and measurement.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Determine if the system has the ability to add, modify, view and delete subscribers and their subscriptions, and whether it is possible for the platform to add, modify, move, view or delete these subscriptions.
3. Document this functionality (or lack thereof).

Stores/Retrieves/Maintains List Of Subscriptions

Purpose: To determine if the platform maintains a list of subscriptions for control and measurement.

Test Steps:

See preceding test above.

Stores/Retrieves/Maintains List Of Fuselets

Purpose: To determine if the platform maintains a list of fuselets for control and measurement.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Determine if the system has the ability to add, modify, view and delete or move fuselets
3. Document this functionality (or lack thereof).

Stores/Retrieves/Maintains List Of Force Templates

Purpose: To determine if the platform maintains a list of force templates for control and measurement.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Determine if the system has the ability to add, modify, view and delete or move force templates.
3. Document this functionality (or lack thereof).

Stores/Retrieves/Maintains List Of Brokers (And Associated Systems Providing Platform Services)

Purpose: To determine if the platform maintains a list of brokers or other providers of system services.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Determine if a list of brokers and other platform centric systems is available.
3. Document this functionality (or lack thereof).

Stores/Retrieves/Maintains List Of Persisted Information Objects

Purpose: To determine if the platform controls a list of persisted information objects.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.

2. Determine if there is a list of persisted information objects and if so, whether a change can be made (moved or replicated) and then stored.
3. Document this functionality (or lack thereof).

Provides Information Relating To Operations (Pub/Sub/Query) Performance

Purpose: To determine if the platform provides operations for monitoring performance statistics.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Determine if the platform can show statistics displaying the number of subscriptions, publications, and queries currently using the JBI as well as other performance metrics.
3. Document this functionality (or lack thereof).

Provides Information Relating To Network Connectivity And Latency

Purpose: To determine if the platform provides network performance statistics and metrics for client related functions.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Determine if the control application displays network connectivity, latency statistics or other performance metrics for clients related functionality.
3. Document this functionality (or lack thereof).

Dissemination Functions

Provide Administrative Querying Capability to <all known> Information Objects

Purpose: To determine if the platform provides querying capabilities for all information objects.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Execute the steps to perform a query of information objects in the JBI.
3. Determine if the query application can see all of the information objects in the platform.
4. Document this functionality (or lack thereof).

Provide Transmission Capability For <all known> Information Objects

Purpose: To determine that the platform provides transmission of information objects to fulfill its primary function of providing published information objects to clients who have

subscribed to the information and provides information to clients who have queried for information.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.
2. Execute the steps to subscribe to information objects utilizing all possible ways of delivery of those information objects.
3. Publish information which fulfills the previously created subscription.
4. Verify that the subscriber receives the subscription in the forms which have been requested. Where interaction is required by the subscribing client, provide this.
5. Query for the same information object(s) and verify delivery of such.
6. Document this functionality (or lack thereof).

Fuselet Management Functions Create Fuselets

Purpose: To determine if the platform supports the creation of fuselets.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.
2. Execute the steps necessary to create a fuselet.
3. Verify that a fuselet has been created and stored using platform services.
4. Document this functionality (or lack thereof).

Modify/Edit Fuselets

Purpose: To determine if the platform supports the modification/edit of fuselets.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.
2. Execute the steps necessary to view, select, modify and re-save a fuselet.
3. Verify that the fuselet has been updated and stored using platform services.
4. Document this functionality (or lack thereof).

Delete Fuselets

Purpose: To determine if the platform supports the deletion of fuselets.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.
2. Execute the steps necessary to view, select, and delete fuselets.

3. Verify that the fuselet has been deleted using platform services.
4. Document this functionality (or lack thereof).

Modify Fuselet Permissions

Purpose: To determine if the platform supports the modification of fuselet permission.

Test Steps:

1. Connect to the yJBI (if not already connected) with provided client software.
2. Execute the steps necessary to view, select, and modify fuselet permissions.
3. Verify that the fuselet has been updated using platform services.
4. Document this functionality (or lack thereof).

Manage/Maintain Fuselets

Purpose: To determine if the platform supports the maintenance/management of fuselets.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Execute the steps necessary to view, select, add, modify, and delete fuselets.
3. Document this functionality (or lack thereof).

Transformation Utility Functions

Provides Legacy System Wrappers

Purpose: To determine if the platform provides any connectivity to legacy systems for connection to the JBI as client using the platform services to provide the capabilities present in the legacy system.

Test Steps:

1. Connect to the yJBI (if not already connected) using a provided legacy system interface designed to explicitly allow interaction with the JBI for the services provided by the connecting system. (For example, if a database server, allowing data to be published from the database into the JBI.)
2. Test the platform services usage from the legacy system of publish, subscribe and query.
3. Document this functionality (or lack thereof).

Provide Exception Handling And Logging.

Purpose: To determine if the program displays errors and/or traps problems such as disappearing clients and stores them in a relevant log.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Execute the steps necessary to subscribe to data from the JBI.
3. Publish information explicitly designed to fulfill the above subscription continuously.
4. After some time pull the network cable.
5. Verify that errors are generated and logged for the incident.
6. Document this functionality (or lack thereof).

Modify/Edit Force Templates

Purpose: To determine if the platform allows modification and/or editing of force templates.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Determine if the controlling interface has the ability to modify and/or edit force templates.
3. Execute any steps to view, select, modify and/or edit force templates.
4. Document this functionality (or lack thereof).

Maintain/Manage Force Templates

Purpose: To determine if the platform has the ability to maintain and/or manage force templates.

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Execute any steps to view, select, modify and/or edit force templates.
3. Determine if the control application can alter the location or set permissions for use of the force templates.
4. Document this functionality (or lack thereof).

Provide Shared Object Capability

Purpose: To determine if the platform has the ability to provide a shared object capability (in support of collaboration tools or for special cases (i.e., the SCR).

Test Steps:

1. Connect to the yJBI (if not already connected) through a provided control interface.
2. Execute any steps to view, select, modify and/or edit specific information objects.
3. Determine if the control application can alter the object to accommodate sharing.
4. Document this functionality (or lack thereof).

4.0 Results

Functional test scoring is based on SAB Report 99/Reference Architecture/FAQ requirements and consists of using a 1 or 0 to denote whether the yJBI did or did not meet a particular requirement, respectively. These are then added and averaged to obtain a final score. This makes the scoring system statistical based and makes the desired aggregate score a 1. This will allow the JBI team to add, amend, or delete functional requirements without having a severe impact on the overall interpretation of the score. It is recommended no more than three significant digits be used for the overall score. Comments will be provided to address particular results of the test.

Document 1-6

TEST TOOLS

In order to facilitate the testing of the JBI prototypes, a number of commercially available tools are available which would help to automate both testing process as well as the requirements generation process. The primary recommendation is for the Rational Test Suite which includes Rational Performance Studio (now called Test Studio) as well as Rational Requisite Pro, Test Manager, etc. There are a number of rationales for this selection, the primary one being that the PACE project (Performance Analysis & Consolidation Engineering Project from AFRL/IFEB) already had a copy of this software and that one of the principles has significant experience with this tool. A lengthy selection process was executed in order to identify a product suitable for this environment by the aforementioned PACE project, and the reader is referred to the studies done by that project for further information.

The next document provides a list of test measurement software that is commercially available or otherwise available which could be used to provide instrumentation for measuring the various metrics necessary to provide the results that the government is seeking in regard to the test goals.

Tools Lists

- AFRL SE Toolkit: A suite of performance measurement programs which monitor and log performance metrics.
- Strace: An open source system call tracer which logs operating system call usage for a process.
- Taztool: A tool which traces disk accesses for analysis of disk activity.
- Memtool: A tool for recording memory utilization for both processes and the operating system.
- lsof: A tool used to log ‘open’ files, allowing analysis of application I/O usage.
- Sysinfo: A tool used to baseline hardware configurations.
- Solaris Resource Manager: An application which allows monitoring and controlling of a Solaris system.
- Ethereal: Open source network analysis and capture tool.
- National Instruments Network Observer: A Tool for capturing, monitoring, and analyzing network traffic.
- Metron Athene: A suite of measurement tools that allow the recording and analysis of metrics for many different systems.
- Rational Test Studio: A suite of performance and load testing tools.
- Mercury Interactive’s Load Runner (and other products): A performance, loading, and functional testing environment.
- BMC’s BGS Best/1 & BMC Patrol: A suite of testing tools allowing for functional, regression, performance, loading, and other tests.
- Teamquest Baseline & Model: A suite of test measurement tools for instrumentation of testing scenarios.
- Aurora Software’s SarCheck: A performance analysis and tuning tool.

- Wily Technology Introscope: An application monitoring test suite and probe (Web, Java, etc.).
- Sybase Diagnostic and Tuning Center: Chronological performance monitoring (associated with Sybase RDBMS).

(This is only a partial list – not intended to be exhaustive or complete.)

This does not include 'built-in' tools of the operating system(s) under test. For example, in Solaris you could use perfmeter, proctool, swap, vmstat, iostat, netstat, lockstat, sar, psr, truss, prtconf, and the process monitoring tools ('p' commands) just to name a few.

Document 1-7

APPLICATION PROFILE

As part of the process of evaluating yJBI prototypes, a necessary step in the providing of an evaluation of their capabilities and characteristics, an application profile needs to be completed in order to assess their impact and accommodate their operational requirements. The following document provides a profile to be used in order to gain a first order approximation of the operational characteristics of the application under test.

Application Profile/Characterization Questions

Application Name and Description:

_____

_____

_____

Identify Application Type (Client-Server, Standalone, Middleware, daemon process).

_____

If it is a daemon process, is it a serving type process, or a function oriented process?

_____

For applications which are Client-Server, Client and Server Parts will be treated as separate applications, so identify as such.

_____

Identify if the application is single Threaded or Multiple Thread capable.

_____

Categorize the application: Online Trx Processing (OLTP), Decision Support Sys (DSS), or, is it a batch processing type program, or monolithic function?

_____

Classify the program as a data storage program, a conduit to/or for data, a processor of data (filter), or strictly performs some function (mathematical or converting).

_____

Does it have multiple main executables? Is it more system than program?

_____

Does the application have multiple processes or is it monolithic?

_____

Is there a minimum hardware recommendation and if so, what is it?

Is there a minimum memory requirement for this application?

Is there a minimum disk space requirement for this application?

Is there a minimum graphics resolution for this application?

Is there a minimum I/O throughput requirement for this application?

Does the application open, close, create, or delete many files during its normal operation?

Is the application network oriented? Does it require network access to function?

Does the application use any of UNIX's piping or semaphore message passing or Streams capabilities?

Does the application make use of any of kernel services provided by the UNIX kernel?

Does the application use any of the built-in capabilities of the shell(s)?

Does the application use any of the external UNIX commands? (AWK, SED, CUT, etc.).

Is the application reentrant?

Is the application interrupt driven?

Does the application have a sleep or interrupt mode of some kind?

Is the application character-based or GUI-based.

If the application is GUI-based, what GUI environment is required? Is GUI provided as part of the application and have special requirements for memory/processor/graphics resolution/etc.?

Is the application graphics intensive? If so, does it require specialized graphic processors, or does use it OpenGL or some other 'X' graphics library?

Does the application depend on user authentication services, or license manager daemons?

Does the application install or require any kind of listening services through TCP/IP ports?

Does the application require NFS server mounts or access via NFS daemons?

Does the application require access to a database on CDROM or some other device? (Including Network or Drives)

What are the data sources for the application (if any)?

Does it use dynamic linking at runtime? If so, identify dependent libraries required for operation:

Does the application support numeric processor calls to the CPU directly or is it dependent on OS for numeric services?

Does the application require other applications to be installed and running in order to operate? (e.g., database services of any kind, network services or daemons of some kind?

Is the application a database or include in its capabilities the ability to create/use private databases of any kind?

If network oriented, what protocol does the program use primarily (TCP/UDP-IP, FTP, Telnet, SNMP, appletack, etc.).

Does it require specific network ports? How many ports does is use?

What language(s) is the application written in (compiled, P-Code, interpreter, etc.) ?

_____

What compiler memory model was used in creating the application (Large Memory, Medium, or Small, etc.)?

_____

Does it have an installation procedure/program? (Does it have an uninstallation procedure/program?)

_____

Are there any prerequisites required for installation or operation in terms of other resident programs, applications, or libraries?

_____

Are there any specific configuration requirements for proper operation or can it use the default settings of the OS?

_____

FUNCTIONAL ATTRIBUTES

List of the application's features and capabilities (functions) (e.g., publisher, subscriber, querying, encryption, mapping, etc.).

1_____

2_____

3_____

4_____

5_____

6_____

7_____

8_____

9_____

10_____

Publish/Subscribe/Query Capabilities or functions?

_____

_____

_____

_____

What Publishing/Subscribing/Querying technology is used?

For Subscribing applications, what are the publishing sources?

What are the query sources for the application?

Is there a limit to the number of subscriptions? What is it?

When publishing data, can the program push data from an external source or does it require access to an internal or connected source?

Can the program save retrieved (published/queried) data locally?

Does the program have an email interface?

What kind of data files can be exported from the application (html, xml, eml, jpg, word docs, etc.)?

Does the GUI interface support drag/drop or cut/paste standards for external sharing of data?

Document 1-8

PERFORMANCE TESTING

In addition to the functional testing mandated by this task, performance and scalability tests were also identified as being relevant. To that end, the following document describes the performance methodology and metrics that would be appropriate for testing the yJBI prototypes. This is followed by the description of the scalability metrics and methodology for testing scalability.

Performance Testing

Performance testing is often used to verify that an application is able to exceed the handling requirements for its particular function (i.e., number of users, amount of transactions, etc.).

Many performance tests that are attempted fail to actually measure application scalability. The reason is because it is difficult to separate application performance and scalability from the underlying operating system and hardware characteristics. Indeed, in many cases it is not possible to separate the application's performance and scalability from the operating system at all, because of the complex inter-relationship between them. However, application and hardware characteristics can usually be differentiated with some effort.

Levels of Performance Testing

A	B	C	D
Performance Profiling Analysis	End-to-End User Response Testing	Algorithm and Code Evaluation	Application Characterization

Classification of performance testing using five scaled models of interaction;

	Very Small	Small	Medium	Large	Huge
No of Clients	10	10-100	50-200	200-1000	1000-10000

In the chart above, showing levels of performance testing, code evaluation and application characterization cannot identify scalability bottlenecks readily. A performance profiling analysis or response time testing can indicate the performance capabilities of the software under test. Both of these tests use scaled models of interaction to test scalability and performance for increasing numbers of clients.

These clients would publish, subscribe, and query a repository containing a library of N objects initially, but growing as the testing was implemented to N times the number of clients plus a random number (seeding based on the number of clients) of information objects. The information objects themselves would vary in complexity (size) in order to simulate real world conditions as closely as possible (size might be determined by selecting randomly from a set of created 'dummy' information objects, with seeding based upon number of available objects).

The simulation would vary the number of publishers, subscribers and queries from a quiescent state (boundary condition), to a state where either the hardware/OS is in a saturation state (boundary condition) or the software either crashes or cannot handle the burden (boundary condition). Particular attention would be paid to the point on the utilization curve where the

software performance begins to degrade in handling the load. (This presumes that the hardware is able to handle the load.)

A goal for the test scenario is to separate repository performance and scalability from the Publish/Subscribe/Query performance and scalability. This will be attempted by replacing the yJBIs repository with a simulation perhaps based upon TestStudio. This may not be possible where the repository is integrated with the publish, subscribe, query engine of the yJBI. A similar goal is to test the repository itself in regard to receiving, delivering and querying information objects. Another goal of the performance testing is to record end to end (as in client) performance of the yJBI system which takes the software and hardware characteristics of all associated equipment and software. Where possible, identify bottlenecks to provide insight into the hardware and software architecture required for optimal performance.

The capabilities of the particular yJBI under test will no doubt impact upon this potential scenario, but where possible, this would be the preferred plan.

A yJBI performance test plan is in development to accomplish these goals. It is expected that unlike the functional test plan, this test plan will have to be significantly changed for each yJBI tested because of different architectures. The PACE (Performance Analysis and Consolidation Engineering) Project's Performance Analysis Methodology (PAM) will be used as a guideline in actually creating the test plan. It is shown here for review:

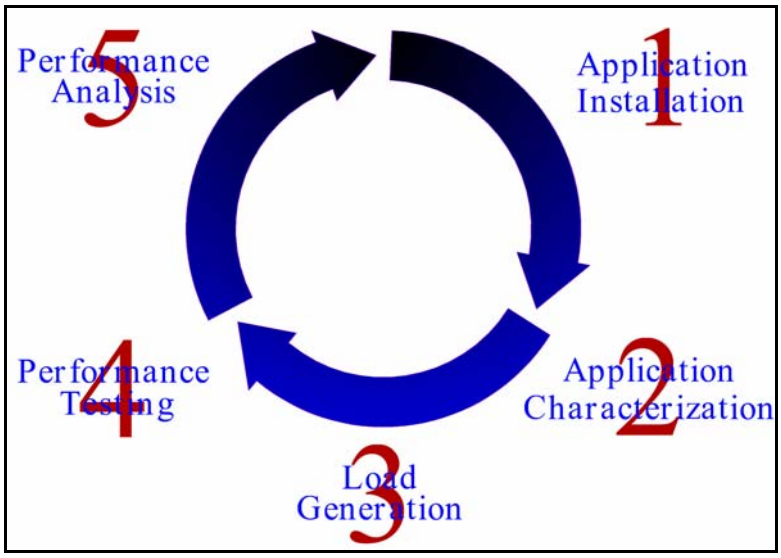

Figure 17: The Performance Analysis Methodology (PAM)

Metrics recorded as part of this testing would include (but not be limited to);

- Time to publish (as in, time from when object is submitted to when it is available to a subscriber) both individual and aggregated statistics

- Subscription response time (time for a published object to be identified as fulfilling a subscription and subscriber notified) for both individual and aggregate statistics (segregated by subscription response requested)
- Query Response time, (time from query request to initial fulfillment of query) both individual and aggregate statistics (segregated by query response requested)
- Number of clients during the tests
- Queries per client
- Query rate per client
- Publications per client
- Publication rate per client
- Subscriptions per client
- Subscription rate per client
- Number of information objects by object type
- Information objects published by client and as a system
- Information object generation rate by client and as a system
- Information objects received (via subscription) by client and as a system
- Information objects received (via subscription) rate by client and as a system
- Information objects received (via query) by client and as a system
- Information objects received (via query) rate by client and as a system
- Unique IO Types Being Published and available in the repository
- Unique IO Types Available For Query and available in the repository
- CPU utilization statistics
- Network utilization statistics
- IO operations statistics
- Operating system statistics (system calls, context switching, interrupts etc.)
- Memory Utilization metrics

Document 1-9

SCALABILITY TESTS

The purpose of scalability tests is to measure whether a particular application or system can continue to deliver its solution or function when the size of the problem is increased. The context of the test scenarios are relevant to the capability that is ostensibly scalable. This document attempts to map out some metrics to use in evaluating systems under test for scalability, as well as scenarios which could be used to identify scalability issues.

Scalability tests measure:

- Requests per second
- Transactions per second
- Kilobytes per second
- Round trip time
- Concurrent connections
- Degradation

Requests per second reports the number of requests or hits a server received for all traffic. It shows the most granular detail of the incidents of interaction between the client and the server. Used in conjunction with other metrics (such as failed requests), to create ratios, requests per second illustrates the scalability of an application. Scalability testing which distinguishes where traffic begins producing significant failures in response to requests is the main goal.

The ***transactions per second*** metric attempts to correlate operational capability (and by inference), overall functionality with the application's throughput capability for specific functionality. Some applications, specifically OLTP (Online Transaction Processing) programs, and other commerce oriented processing systems define operations as being transaction oriented and are easily characterized in this manner. Other systems do not have clear definitions of what constitutes a transaction and for these this metric is not necessarily useful. However, when the transaction per second metric begins to decrease even though the number of clients/users is increasing, the system is experiencing a degradation in scalability.

Kilobytes per second identifies raw throughput capabilities of an application. This metric is useful in scalability studies because when the kilobytes per second does not increase in relation to the number of users or clients then you have reached a scalability restriction. This metric can be misleading at times because the functionality of a complex system may not translate into greater raw data throughput for a larger number of transactions.

Round trip time measures the end-to-end performance of a system. It includes the time for every layer and component in the chain between the client user and the application server and back. This is the client or user view of performance and scalability for an application. This often also includes a response time measurement for connection and login as a separate metric.

Concurrent connections is the number of simultaneous connections to a server for particular system. When connections are increasing but the requests per second are remaining nearly constant is an indication that connections are being required to stay open longer to service the

requests. Like kilobytes per second, this metric can be misleading at times because of differing conditions per connection. (For example: connections being maintained over low-bandwidth links versus high-bandwidth short duration connections.) Also, client transaction types may also be varying by connection, so that some connections represent different behavior.

Degradation is where the relationship between increasing activity as being measured by the metrics described above and resource allocation has ceased being proportional so that fewer resources are being used for more activity. Degradation occurs where the 'bend in the knee' of the curve of utilization occurs.

In order to measure these metrics and thereby conclude whether a tested system is scalable, there are number of factors which impact on these metrics and they are documented here.

Size Matters

Size does matter. The amount of data being transferred back and forth between client and server is the single largest factor when measuring overall throughput for any application. The larger the data content, the slower everything goes. In truth, things are not moving more slowly, there is just more to move and so this takes longer. A useful analogy from the hardware world is the relationship between drive performance and overall computer speed. No system is better than its worst part. In computers, the single greatest factor in their performance is the throughput to the drive. In the client/server environment, the size of data being sent and received is the greatest factor in determining throughput. Throughput is not scalability however.

Throughput impacts on scalability and its measurement because where content is changing (in terms of amount); resources are being used in proportional relationship to the size of that content. It is extremely important to measure the throughput (raw performance – kilobytes per second) in relation to the functionality (transactions per second) being delivered and varying the throughput explicitly to see the affect of resource allocation for 'raw' throughput versus functional allocation of resources to deliver capability. In other words, separating behavior associated with just 'moving the data' versus the function or operation being applied to/for that data.

Having identified metrics and factors affecting the tests, the next section delineates a method for creating test scenarios to test for scalability.

Scalability Testing Methodology

The simplest test for scalability involves increasing the number of items that a system has to handle (process, transmit, etc.). A more sophisticated test would vary the number of items and their size as a function. Using the publish, subscribe, and query paradigm, a useful first test would be to measure the individual function's scalability independently and then measure their scalability as a varying matrix of possibilities. It is important to identify the relationships between the possible permutation of implement-able test functions of application and their real world counterparts. It is not valuable to test for permutations which will never occur in the real world except to establish boundary conditions.

Where possible, analysts should identify real world worst case conditions for the application(s) under test and strive to test for these scenarios. These scenarios provide a valuable starting point

for testing scalability, but the painful lesson from the real world is that the actual usage will be *at least* twice as much traffic then worst scenario as a realistic starting point.

Document 1-10

CONCLUSION

Systems being developed today can be engineered using disparate technologies and this has a significant effect on the scalability and performance capabilities of the end product. One of the primary goals of the evaluation is to provide feedback about the suitability of various technologies in relation to their capabilities for the JBI program. A performance and scalability test of yJBI prototypes is the principle way which this goal can be met. These documents map the process to follow in evaluating specific yJBI prototypes and measuring their individual capabilities with respect to their performance and scalability under varying test conditions. An individual yJBI prototype which would have been appropriate was not available for testing during the execution of this work. It is anticipated that these processes will be used to evaluate a yJBI prototype when one does become available.